2020 자율주행 교육

WeGo 위고 주식회사

목차

01

Yolov3 Train

WeGo

01. Yolov3 Train

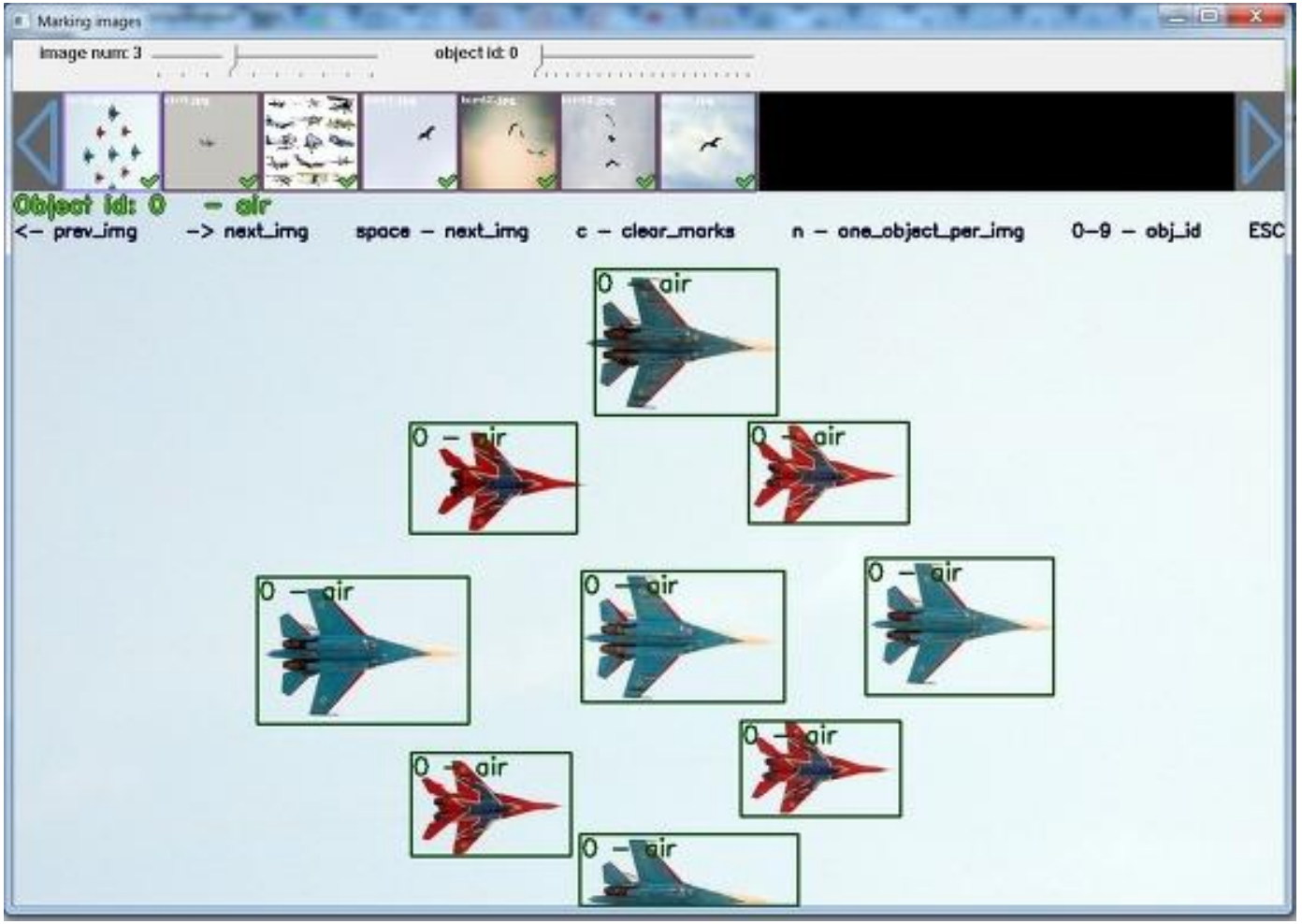
- Yolov3 에 새로운 Data를 이용하여 Training
- Yolo 학습에는 YOLO_MARK가 필요

→ YOLO Mark는 이미지 파일에 직접 Bounding Box를 그려주는 Tool

- git clone https://github.com/AlexeyAB/Yolo_mark
- cd Yolo_mark
- cmake .
- make
- sudo chmod +x linux_mark.sh
- ./linux_mark.sh

01. Yolov3 Train

- 출력된 결과를 통해, Object의 개수와 Image의 개수를 확인할 수 있음



01. Yolov3 Train

- Yolo_mark/x64/Release/data/img
- 위 경로에 Training을 시킬 Image를 저장
- 이 후, Yolo_mark/x64/Release/data의 obj.data를 열어 Classes의 개수를 변경
- obj.names에서 class의 이름을 지정
- 이후 다시 ./linux_mark.sh를 실행하여, Bounding Box를 직접 넣어서 Labeling
- Labeling을 진행하면, image와 이름이 같은 .txt파일이 생성 (Box의 좌표값 및 class)
- yolo-obj.cfg 맨 아래의 classes 를 위와 같은 숫자로 수정
- 그리고 마지막 convolutional layer의 filters를 5 x (class수 + 5)로 수정

WeGo

01. Yolov3 Train

- http://pjreddie.com/media/files/darknet19_448.conv.23
- 위 경로를 통해 convolutional layer를 설치 (darknet directory에 저장)
- x64 폴더를 그대로 darknet 폴더로 전체 복사
- 복사한 x64 폴더 내부의 obj.data의 경로를 복사된 경로를 기준으로 재설정

→ 기존의 data/train.txt와 같은 형태를 x64/Release/data/train.txt 형태로 변경

- 이후 ./darknet detector train data/obj.data yolo-obj.cfg darknet19-448.conv.23
- Training이 진행된다.

01. Yolov3 Train

- Training이 진행되며, 왼쪽 아래의 숫자가 Training이 진행된 횟수를 의미
- 724.731750는 loss function의 평균을 의미
- loss function의 값이 0.xx와 같이 감소할 때까지 지속해서 학습 진행
- 중간 중간마다 backup 폴더에 weights 파일이 생성되는 것을 확인할 수 있음

```
mask_scale: Using default '1.000000'
Loading weights from /home/enundeep1/Martin/DataSet/ski/yolov2_ski.23...Done!
Learning Rate: 0.001, Momentum: 0.9, Decay: 0.0005
Resizing
608
Loaded: 2.611784 seconds
Region Avg IOU: 0.142211, Class: 0.254498, Obj: 0.762167, No Obj: 0.556610, Avg Recall: 0.000000,  count: 25
Region Avg IOU: 0.110679, Class: 0.362280, Obj: 0.722342, No Obj: 0.555096, Avg Recall: 0.045455,  count: 22
Region Avg IOU: 0.159554, Class: 0.258459, Obj: 0.697720, No Obj: 0.554395, Avg Recall: 0.000000,  count: 21
Region Avg IOU: 0.187189, Class: 0.190858, Obj: 0.620841, No Obj: 0.554991, Avg Recall: 0.095238,  count: 21
Region Avg IOU: 0.117658, Class: 0.281086, Obj: 0.732644, No Obj: 0.554893, Avg Recall: 0.000000,  count: 54
Region Avg IOU: 0.188215, Class: 0.274354, Obj: 0.666201, No Obj: 0.558286, Avg Recall: 0.121951,  count: 41
Region Avg IOU: 0.232358, Class: 0.211601, Obj: 0.655876, No Obj: 0.556980, Avg Recall: 0.080000,  count: 25
Region Avg IOU: 0.201465, Class: 0.270627, Obj: 0.667707, No Obj: 0.555929, Avg Recall: 0.111111,  count: 18
1: 724.731750, 724.731750 avg, 0.000000 rate, 10.786442 seconds, 64 images
```

WeGo

01. Yolov3 Train

- 중간 중간 저장되는 Weight의 주기를 변경하고 싶을 경우

→ darknet/examples/detector.c 파일 내부의 train_detector 함수 조정

→ 130 ~ 146번째 라인의 값을 수정하여, 저장될 반복 횟수 등을 조정

```
130         if(i%100==0){
131 #ifdef GPU
132             if(ngpus != 1) sync_nets(nets, ngpus, 0);
133 #endif
134             char buff[256];
135             sprintf(buff, "%s/%s.backup", backup_directory, base);
136             save_weights(net, buff);
137         }
138         if(i%10000==0 || (i < 1000 && i%100 == 0)){
139 #ifdef GPU
140             if(ngpus != 1) sync_nets(nets, ngpus, 0);
141 #endif
142             char buff[256];
143             sprintf(buff, "%s/%s_%d.weights", backup_directory, base, i);
144             save_weights(net, buff);
145         }
146         free_data(train);
```

01. Yolov3 Train

- 최종 결과 확인을 위해 Test 진행
- ./darknet detector test data/obj.data yolo-obj.cfg backup/yolo-obj_x00.weights data/<image file>
- https://viewingcat.tistory.com/1255 의 사이트를 참고하여, Fatkun을 이용한 구글 이미지 일괄 다운로드 진행 가능
- 또한 https://github.com/AlexeyAB/darknet을 참고하여, training 시 사용할 수 있는 유용한 툴 및 사용할 Weight 및 추천 Training 횟수 확인 가능
- Class 수 x 2000이 권장 Training 횟수

WeGo